Seeding and Bezier Tracking in LArSoft

Ben Jones

1. Seed Finding Updates

Reminder: seeds are formed by short groups of spacepoints with a strong directionality.

They seek parts of tracks where there is an unambiguous track direction in 3D

Parameters are:

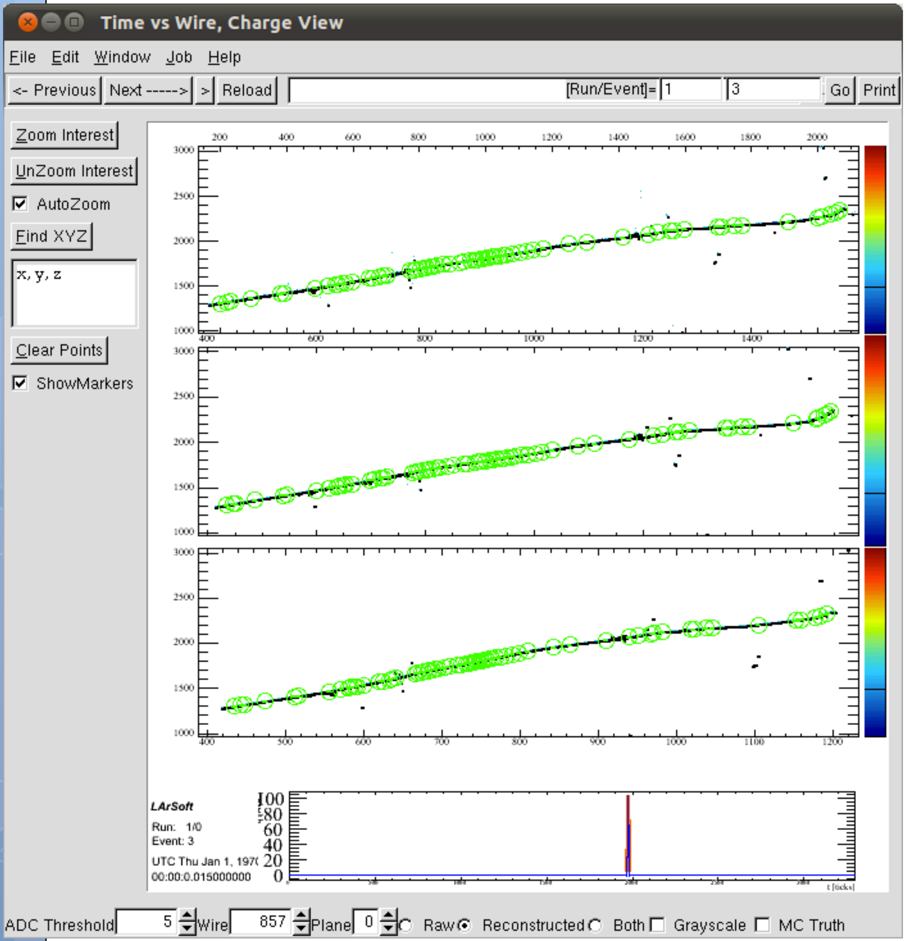
Filter, Merge	- SpacePointService parameters
SeedMode	- 0 (find 1 seed) 1 (find many seeds)
SeedLength	- in cm
MinPointsInSeed	- this many points within SeedLength cm
AngularDev	- SD of seed angular deviation from spine
Source	- 0 (from cluster combos) 1(from bare hits)
HitModuleLabel, ClusterModuleLabel	– data products

Improvements since Last Time

- Improved direction finding – much more robust at finding seed segments
- Can be fed on plain hits (now default) as well as cluster combinations
- New geometrical methods:
  - Seed->GetAngle(Seed AnotherSeed)
  - Seed->GetDistance(Seed AnotherSeed)
  - Seed->GetProjAngleDiscrepancy(Seed AnotherSeed)
  - Seed->GetProjDiscrepancy(Seed AnotherSeed)
- Two new event display views

old

new



[Run/Event]= 1 3

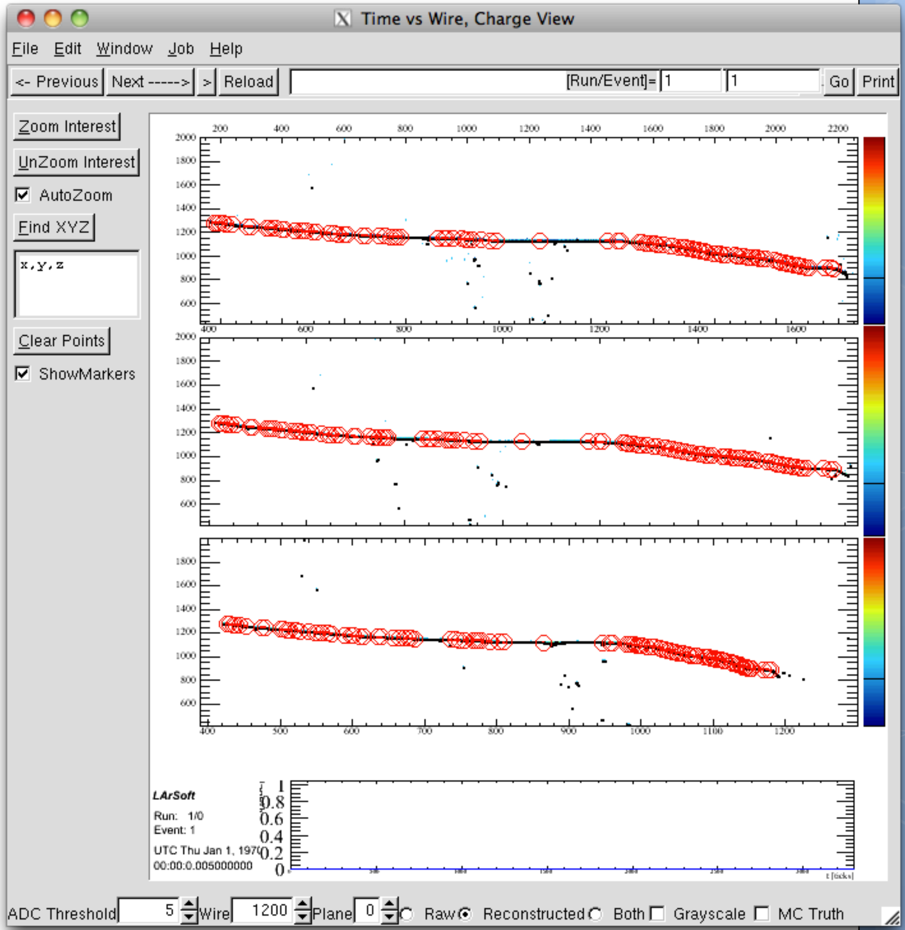
☑ AutoZoom

x, y, z

☑ ShowMarkers

LArSoft
Run: 1/0
Event: 3
UTC Thu Jan 1, 197(
00:00:0.015000000

ADC Threshold 5 Wire 857 Plane 0 Raw ● Reconstructed ○ Both □ Grayscale □ MC Truth



[Run/Event]= 1 1

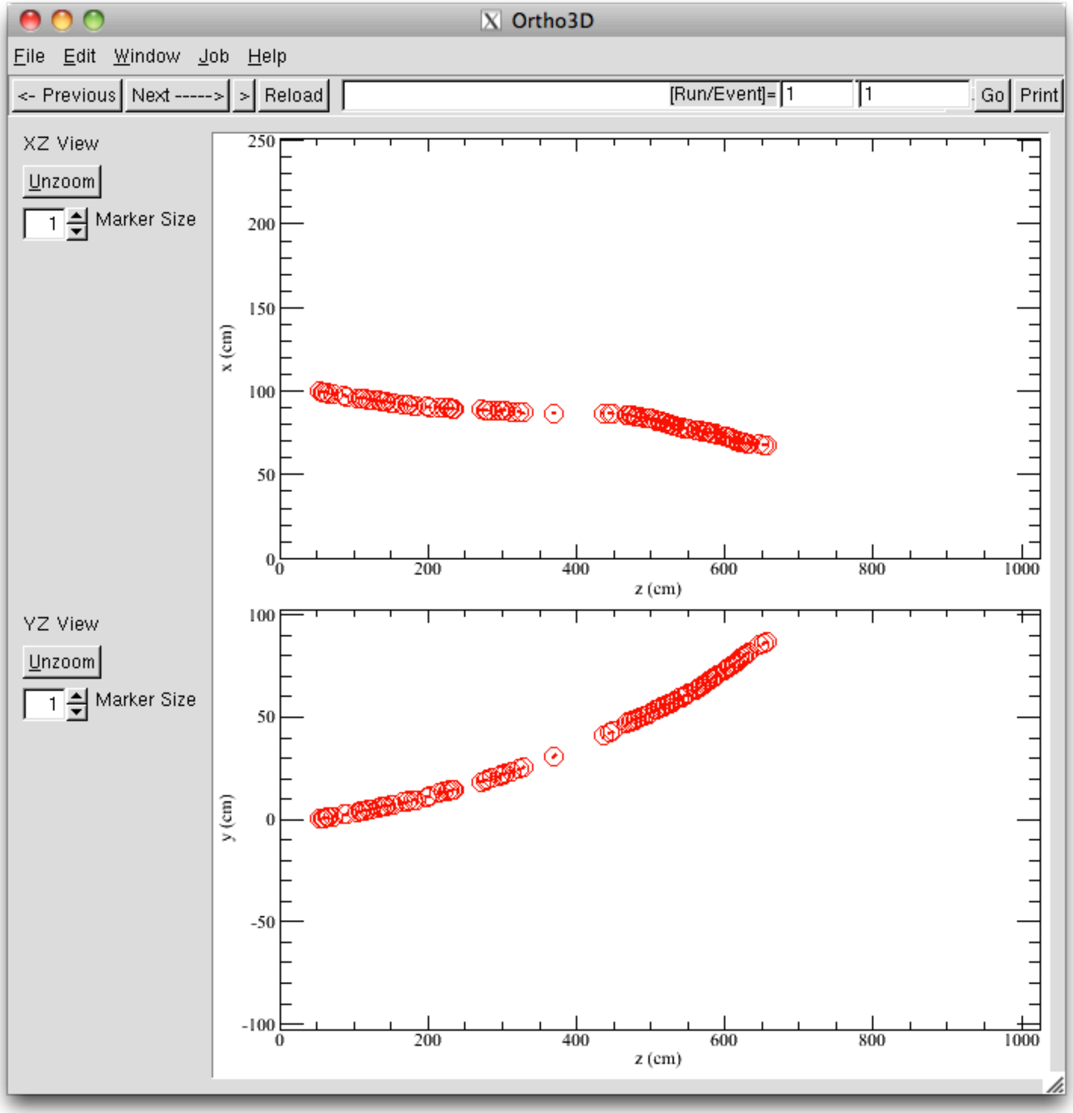
☑ AutoZoom

x, y, z

☑ ShowMarkers

LArSoft
Run: 1/0
Event: 1
UTC Thu Jan 1, 1970
00:00:0.005000000

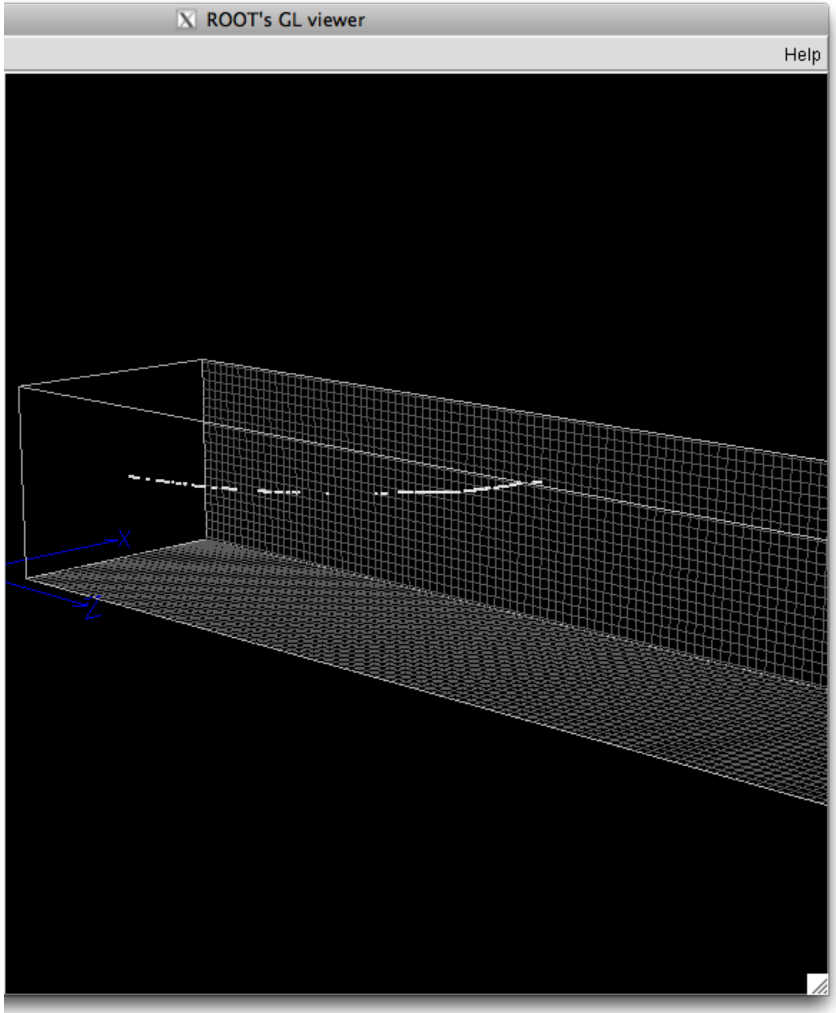
ADC Threshold 5 Wire 1200 Plane 0 Raw ● Reconstructed ○ Both □ Grayscale □ MC Truth

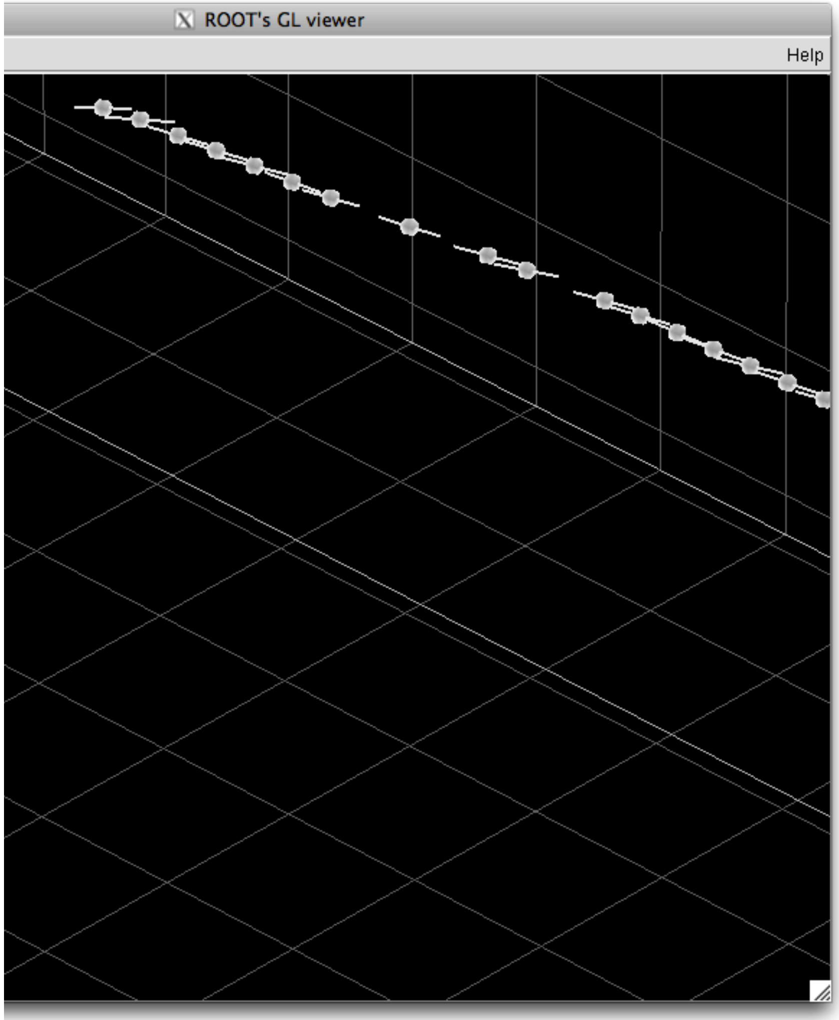


New evd views for Seeds:

This slide: Ortho3D

Next slide: 3D view

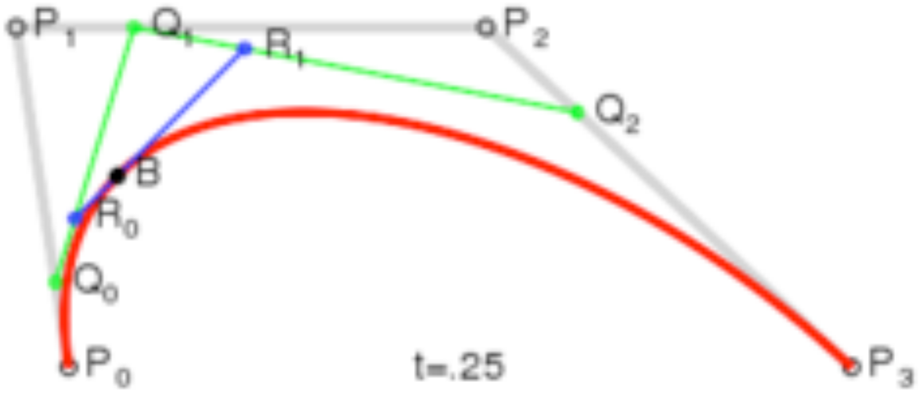

ROOT's GL viewer
Help
X
Z


ROOT's GL viewer
Help

Bezier Tracks

- In multi-seed mode, seeds define a clear pathway for 3D track
- By connecting them with 3rd order Bezier curves we get a smoothly parameterized track
- This object is called a BezierTrack, and can be stored in the event as a set of points and directions (one for each seed) as a recob::Track
- Each segment has a seed at each end and is a continuously parameterized function in 3D
- The segments are in fact totally hidden from the end user (next slides)

Bezier Curve Segments



Curve matches seed point and direction perfectly at each seed, and varies continuously between them

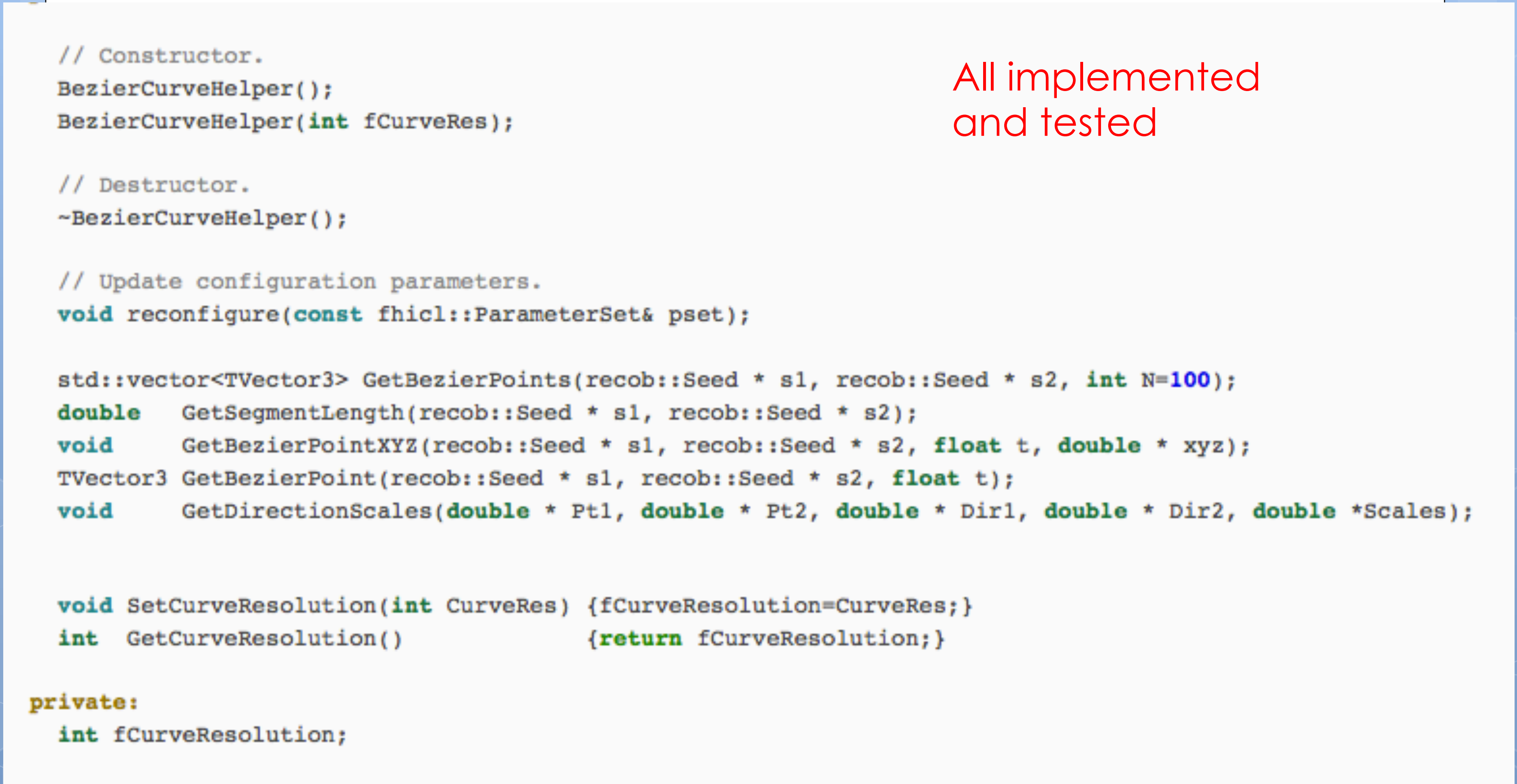
$R(s) = s^3\, P_0 + (1\text{-}s)s^2\, P_1 + (1\text{-}s)^2 s\, P_2 + (1\text{-}s)^3\, P_3$

P_0 and **P_3** are the seed points
P_1 and **P_2** are the seed points + seed direction *(some scale)

The scale is set such that $|\mathbf{P_3 - P_2}| = |\mathbf{P_2 - P_1}| = |\mathbf{P_3 - P_0}|/4$

Local Operations on Bezier Curves

- Since it is a continuous function, can ask for position, curvature, direction, etc at any point
- Each segment is parameterized by 0<s<1
- Lengths etc are not calculated analytically, but rather numerically by dividing curve up and summing displacements – hence always do length calculation with some finite resolution
- Operations to find positions along a Bezier segment are performed by a helper object called BezierCurveHelper in TrackFinder

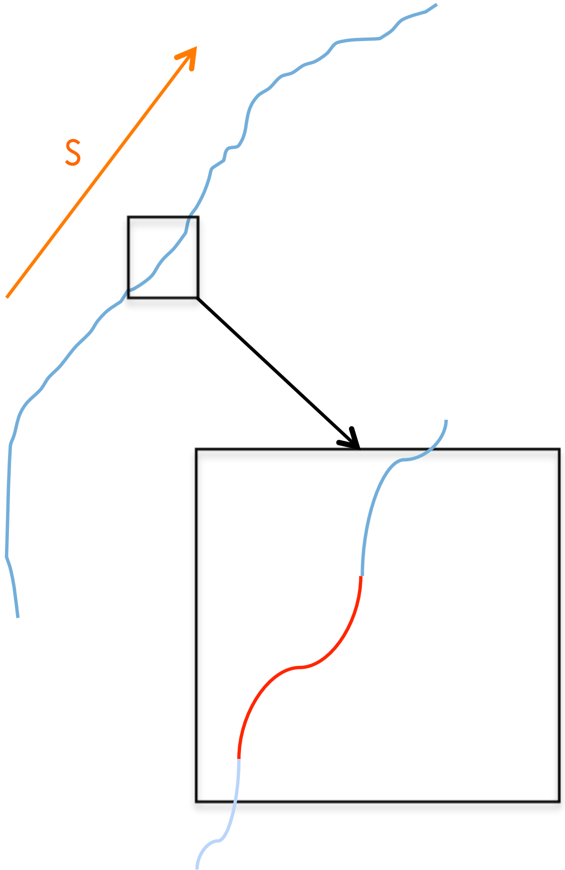


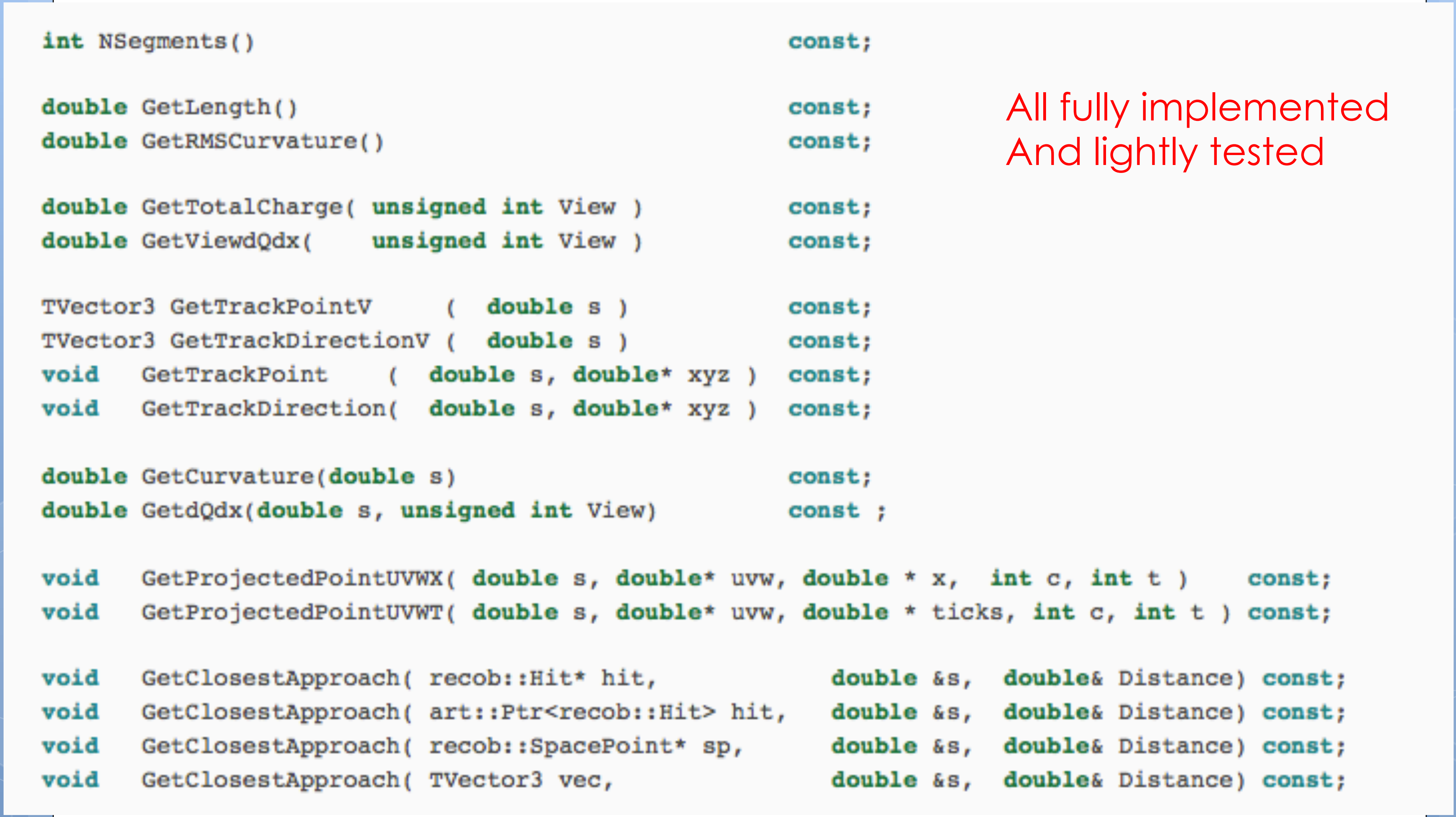
```
    // Constructor.
    BezierCurveHelper();
    BezierCurveHelper(int fCurveRes);

    // Destructor.
    ~BezierCurveHelper();

    // Update configuration parameters.
    void reconfigure(const fhicl::ParameterSet& pset);

    std::vector<TVector3> GetBezierPoints(recob::Seed * s1, recob::Seed * s2, int N=100);
    double   GetSegmentLength(recob::Seed * s1, recob::Seed * s2);
    void     GetBezierPointXYZ(recob::Seed * s1, recob::Seed * s2, float t, double * xyz);
    TVector3 GetBezierPoint(recob::Seed * s1, recob::Seed * s2, float t);
    void     GetDirectionScales(double * Pt1, double * Pt2, double * Dir1, double * Dir2, double *Scales);


    void SetCurveResolution(int CurveRes) {fCurveResolution=CurveRes;}
    int  GetCurveResolution()             {return fCurveResolution;}

  private:
    int fCurveResolution;
```

All implemented
and tested

The Whole as the Sum of its Parts

- User never needs to know about Bezier segments since they are hidden
- On construction, the track works out the length of every segment and the length of the track, and so figures out how to apportion a global s value, 0<s<1 along the entire track, between segments
- User can say GetTrackPoint(0.25) to find the point which is ¼ way along the entire track
- Likewise GetDirection(s), GetCuvature(s), etc



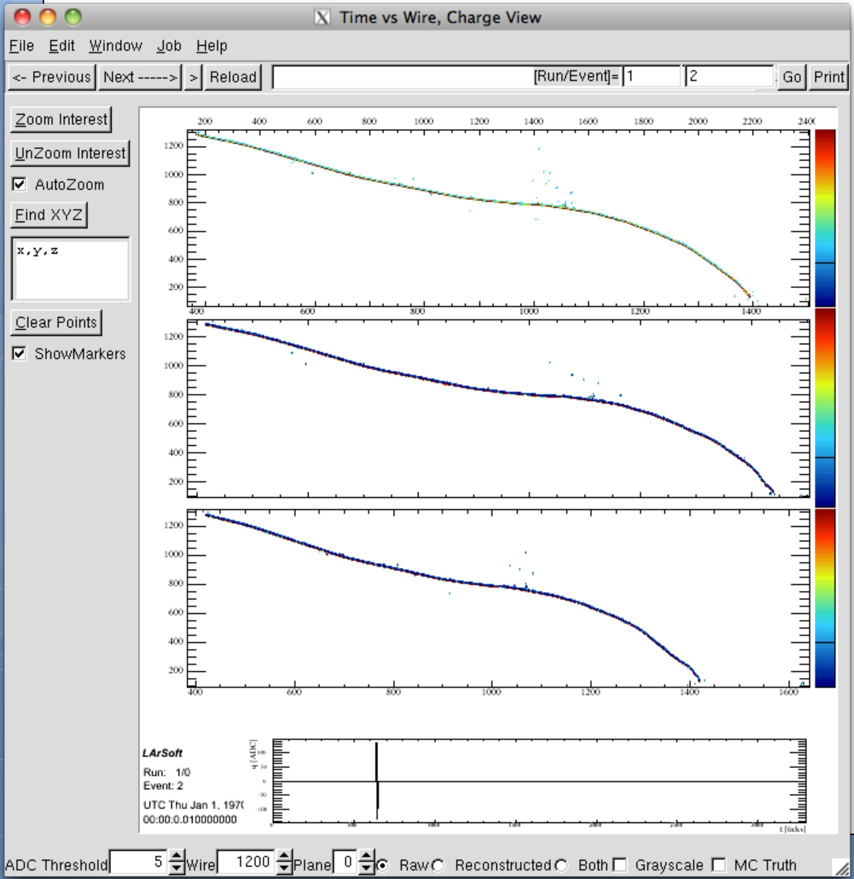


```
int NSegments()                                     const;

double GetLength()                                  const;
double GetRMSCurvature()                            const;

double GetTotalCharge( unsigned int View )          const;
double GetViewdQdx(    unsigned int View )          const;

TVector3 GetTrackPointV      (  double s )          const;
TVector3 GetTrackDirectionV (  double s )          const;
void    GetTrackPoint     (  double s, double* xyz )  const;
void    GetTrackDirection(  double s, double* xyz )  const;

double GetCurvature(double s)                       const;
double GetdQdx(double s, unsigned int View)         const ;

void    GetProjectedPointUVWX( double s, double* uvw, double * x,  int c, int t )    const;
void    GetProjectedPointUVWT( double s, double* uvw, double * ticks, int c, int t ) const;

void    GetClosestApproach( recob::Hit* hit,              double &s,   double& Distance) const;
void    GetClosestApproach( art::Ptr<recob::Hit> hit,     double &s,   double& Distance) const;
void    GetClosestApproach( recob::SpacePoint* sp,        double &s,   double& Distance) const;
void    GetClosestApproach( TVector3 vec,                 double &s,   double& Distance) const;
```

All fully implemented
And lightly tested

Bezier Track in evd

Raw



[Run/Event]= 1 2

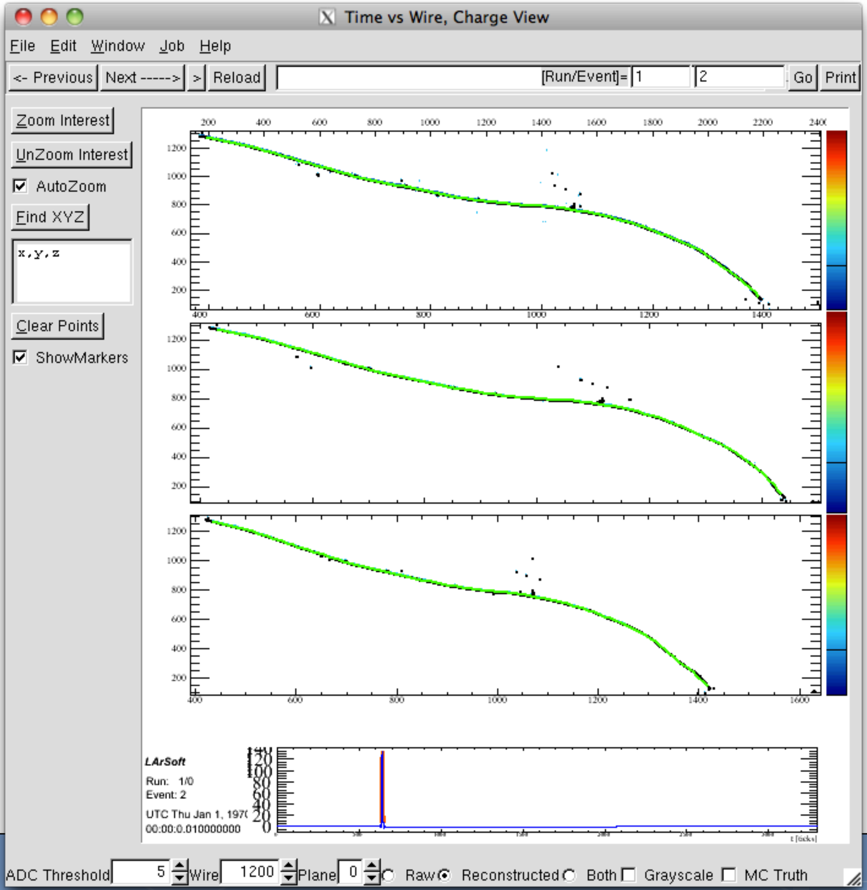
☑ AutoZoom

☑ ShowMarkers

LArSoft
Run: 1/0
Event: 2
UTC Thu Jan 1, 197(
00:00:0.010000000

ADC Threshold 5 Wire 1200 Plane 0 ● Raw ○ Reconstructed ○ Both ☐ Grayscale ☐ MC Truth

Reconstructed 3D track



[Run/Event]= 1 2

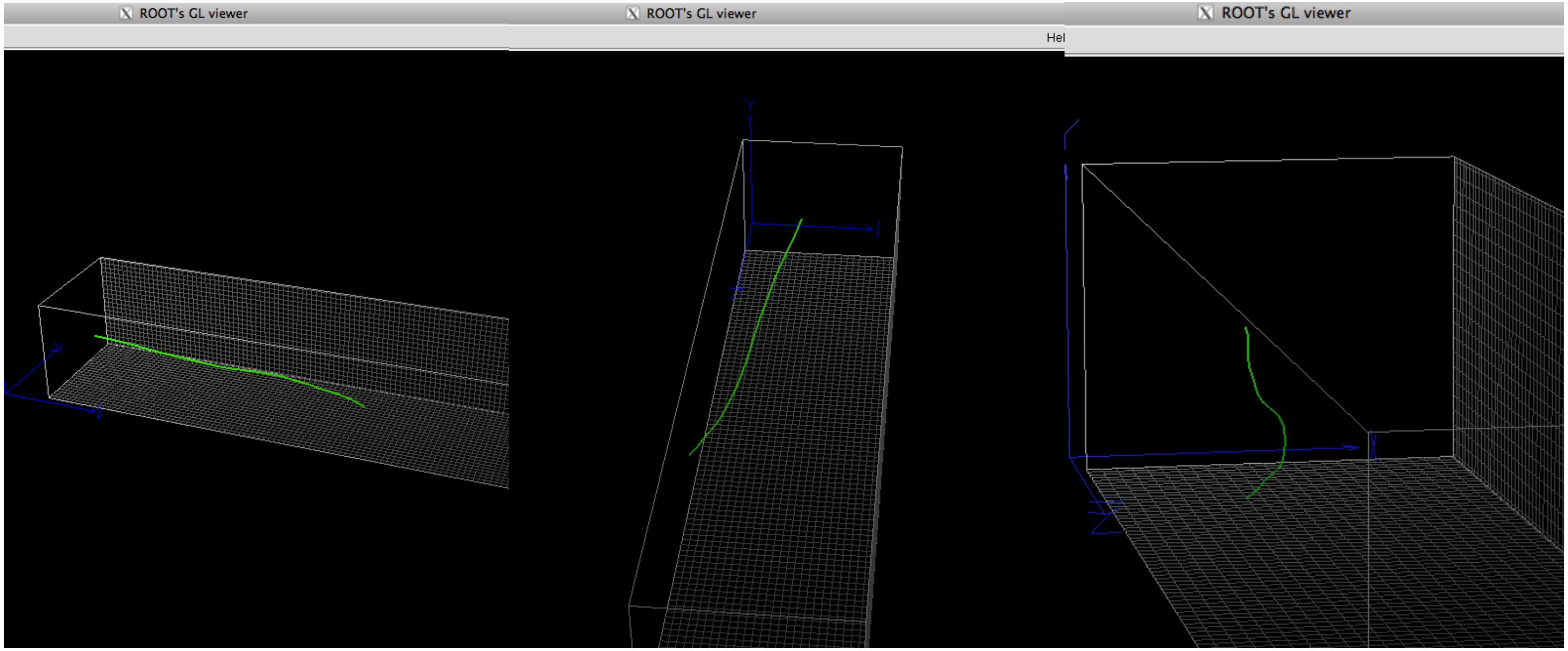
☑ AutoZoom

☑ ShowMarkers

LArSoft
Run: 1/0
Event: 2
UTC Thu Jan 1, 197(
00:00:0.010000000

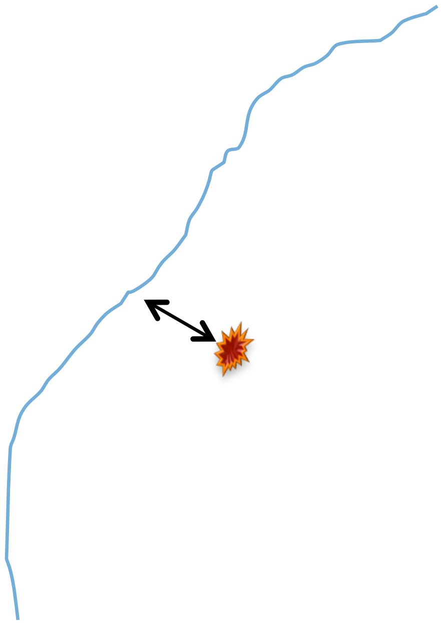
ADC Threshold 5 Wire 1200 Plane 0 ○ Raw ● Reconstructed ○ Both ☐ Grayscale ☐ MC Truth

Bezier track in 3D

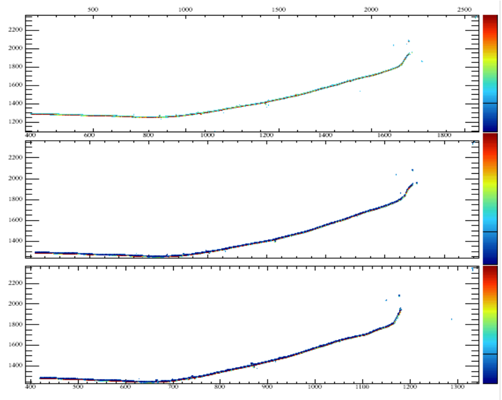


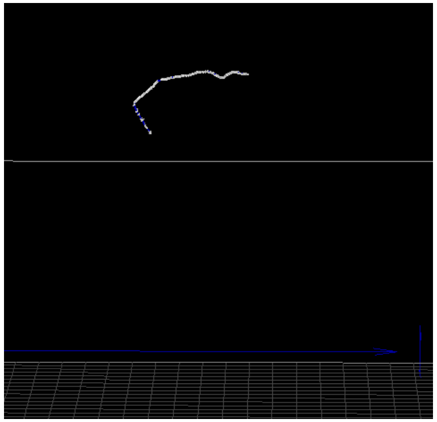
Physics Methods

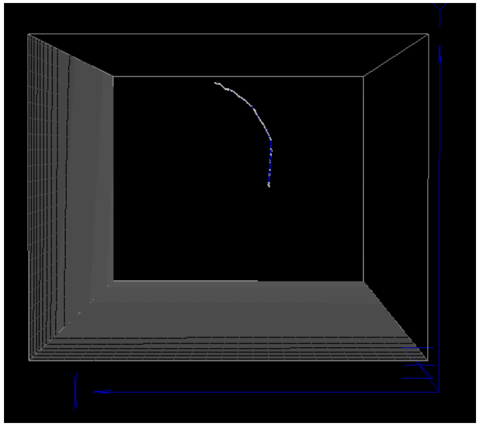
- BezierTracker takes track and collects all nearby hits in each view, using GetClosestAproach method
- Local track pitch in 3D is known at every point, so fully pitch corrected dQdx in 3D is stored for each track segment and for each view
- Easy to go from this to an average per view also
- We also know the track curvature smoothly along the trajectory, so RMSCurvature along the track can be calculated

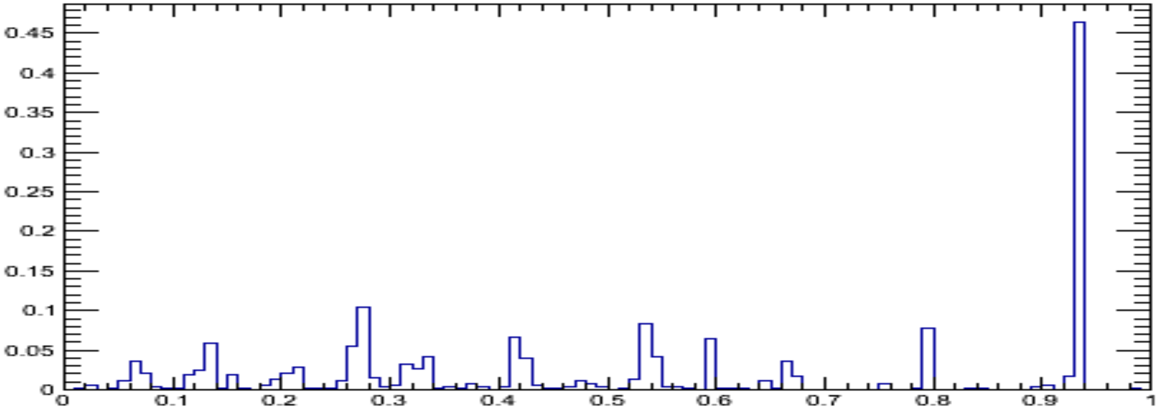


Curvature along track



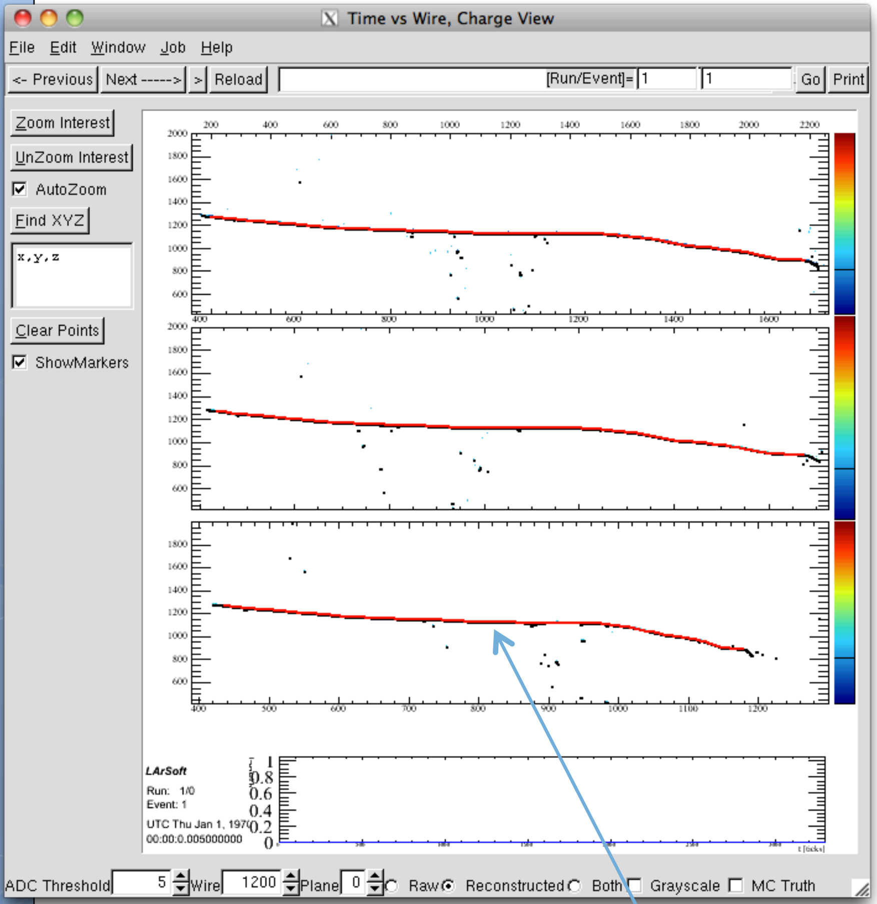






Kinks much easier to see end-on!

dQdx along track in each view…



[Run/Event]= 1 1

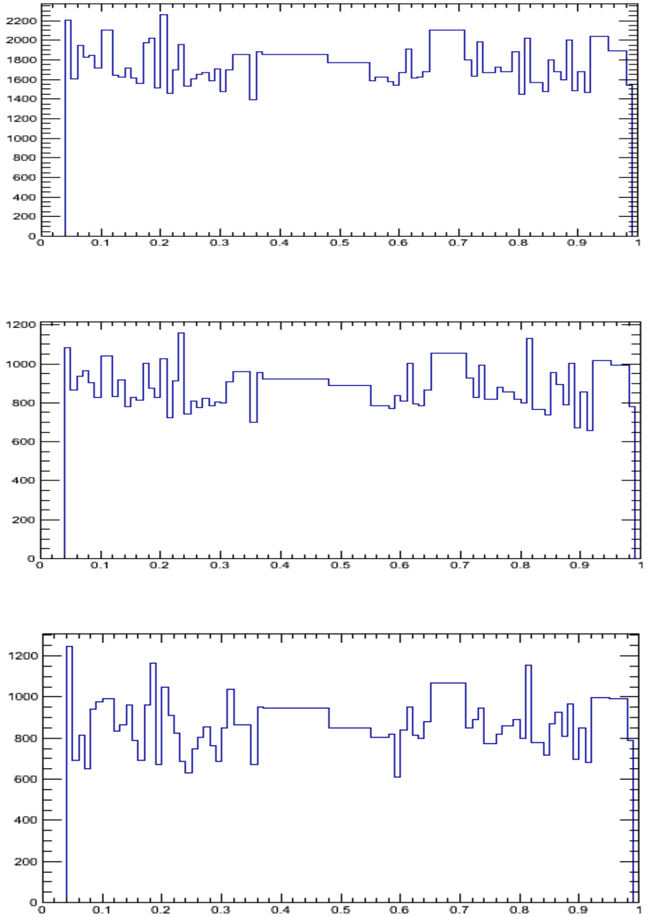
☑ AutoZoom

x,y,z

☑ ShowMarkers

LArSoft
Run: 1/0
Event: 1
UTC Thu Jan 1, 1970
00:00:0.005000000

ADC Threshold 5 Wire 1200 Plane 0 ○ Raw ● Reconstructed ○ Both ☐ Grayscale ☐ MC Truth



This would be very very hard with spacepoints alone

Coming up

- Multi track events (narrowly missed the cut for today)
- Verification against mc-truth information
- Tuning of track fitting parameters